STRUCTURES

(PART 1)

prepared by Senem Kumova Metin
modified by İlker Korkmaz

“struct” concept

- A **structure** has components, called ***members***, which can be of various types.

- The ***declaration of a structure*** captures the information needed to represent the related structure.
- The keyword is ***struct***

Structures (I)

provide the developers to aggregate variables of different types

- EXAMPLE: You have a file including the records below:

ID	NAME	GRADE
12	Joe	A
17	Mary	B
89	Sue	A
10	Tom	C

```
int id[4];   char *name[4];   char grade[4];
id[0]=12;  name[0]="Joe";    grade[0]='A';
```

Structures (II)

"record" is the tag name

"struct record" is the new type

```
struct record {
    int ID;
    char * name;  // char name[25];
    char grade;
};

int main(){
    struct record s1;
    struct record s2;
    s1.ID=12; s1.name="Joe"; s1.grade='A';
    s2.ID=17; s2.name="Mary"; s2.grade='B';
    // ...
}
```

ID	NAME	GRADE
12	Joe	A
17	Mary	B
89	Sue	A
10	Tom	C

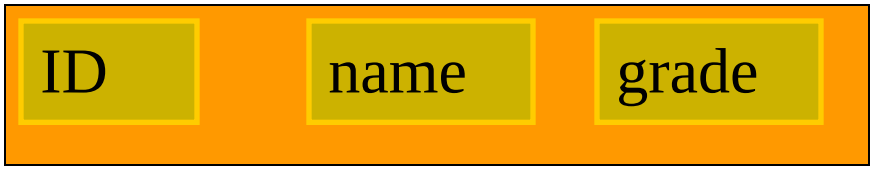
Structures (III)

```
struct record { int ID; char * name; char grade; };
//alternative: struct record { int ID; char name[25]; char grade; };
struct record s1;
```

s1


```
struct record s2;
```

s2

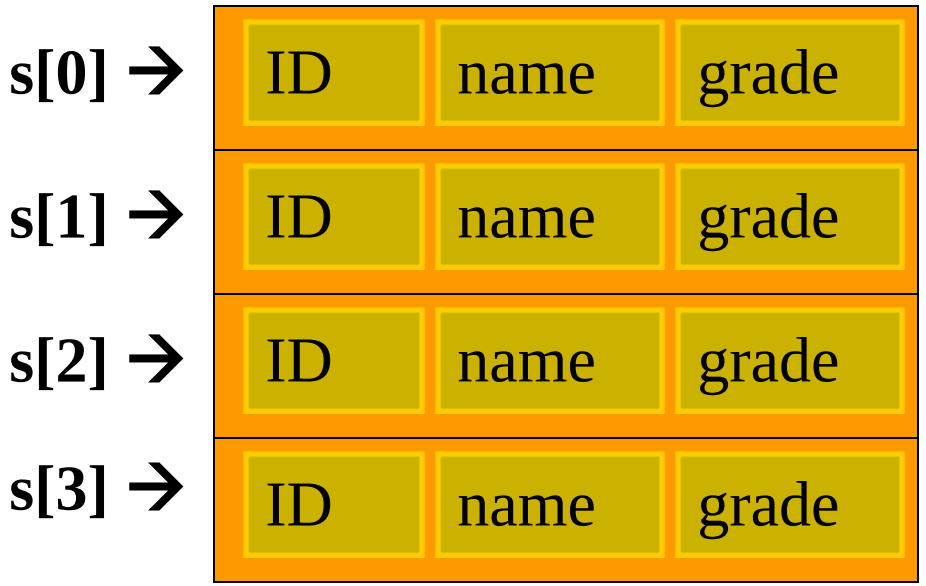


Structures (IV)

```
struct record {
    int ID;
    char * name;
    char grade;
};

struct record s[4];

s[0].ID=12;
s[0].name="Joe";
s[0].grade='A';
s[1].ID=17;
s[1].name="Mary";
s[1].grade='B';
// What does the following mean?
s[2].grade=s[0].grade;
// What about the following?
s[3]=s[1];
```



ID	NAME	GRADE
12	Joe	A
17	Mary	B
89	Sue	A
10	Tom	C

Structures : DECLARATION ALTERNATIVES (I)

Declaration 1 :

```
struct record { int ID; char * name; char grade; };
 // struct record { int ID; char name[25]; char grade; };

struct record s1; // sample definitions
struct record s2;
struct record s3;
```

Alternative Declaration 2 :

```
struct record {  int ID; char * name; char grade; } s1, s2;

struct record s3; // sample definition
```

Structures : DECLARATION ALTERNATIVES (II)

Alternative Declaration 3 :

```
struct {  int ID; char * name; char grade; }  s1, s2;

/* no tag name */
/* no permission to declare other variables of this type */
```

Structures : DECLARATION ALTERNATIVES (III)

Alternative Declaration 4 :

```
struct record {  int ID; char * name; char grade; };
typedef struct record rec;

rec s1; // sample definitions
struct record s2;
```

Alternative Declaration 5 :

```
/* high degree of modularity and portability */

typedef struct {  int ID; char * name; char grade; } rec;

rec s1; // sample definitions
rec s2;
```

INITIALIZATION (I)

```
struct Info {
    char name[10];
    int length;
    int weigth;} men[3]= {"Tom",180,65,
                          "George", 170,68,
                          "Bob", 190,100};
-----------------------------------------------

struct Info women[2]={{"Mary", 170, 55}, {"Sue",
    160,67}};

-----------------------------------------------
struct Info myInformation;
myInformation.name="Jane";
myInformation.length=160;
myInformation.weigth=50;
```

INITIALIZATION (II)

```
#include<stdio.h>
struct PhysicalDetails {
    int length;
    int weigth;
} ;

struct EmployeeRecord {
    int salary;
    int workingHour;
    struct PhysicalDetails phyInfo;
    /* a struct may include (an)other structure(s)*/
} ;

int main(){
    struct EmployeeRecord s1;
    s1.salary=10000;
    s1.workingHour= 6;
    s1.phyInfo.length=180;
    s1.phyInfo.weigth=78;
    // ...
}

/* struct EmployeeRecord b[2] = { { 10000, 7, {190, 78} },
                                  { 15000, 8, {180, 60} } };      */
```

STRUCTURES AS FUNCTION ARGUMENTS (I)

```
#include<stdio.h>

struct date { int day; int month; int year; };
struct date calculate ( struct date d);

int main(){
      struct date today ={10, 12, 2014};
      struct date nextDay;
      nextDay= calculate(today);
      printf("The next day is        %d.%d.%d",
              nextDay.day,nextDay.month,nextDay.year);

      return 0;
}

struct date calculate ( struct date d){
      struct date n;
      if( d.day==30)
              if(d.month==12)
                        { n.day=1; n.month=1; n.year=d.year+1; }
              else
                        { n.day=1;n.month=d.month+1; n.year=d.year;}

      else
              {n.day=d.day+1; n.month=d.month; n.year=d.year; }

      return n;
}
```

STRUCTURES AS FUNCTION ARGUMENTS (II)

```
#include<stdio.h>

struct student { char name[25]; int ID; int grade; };
double average ( struct student x[], int size) ;  // returns the average of grades
typedef struct student st;

int main() {

     st class[30];  int i;

     for(i=0;i<30;i++)
          scanf("%s%d%d", class[i].name, &class[i].ID, &class[i].grade);

     printf ("%.2f", average(class,30) ); // average(&class[0],30)

     return 0;
}

double average ( struct student x[], int size) {
            /* can also be written as
                        double average ( struct  student * x, int size)
                              OR
                        double average ( st * x, int size) */
     double a=0.0; int i;
     for(i=0;i<size;i++)
         a= a+x[i].grade; // a= a+(*(x+i)).grade;
     a=a/size;
     return a;
}
```

STRUCTURES AS FUNCTION ARGUMENTS (III)

```
#include<stdio.h>

struct complex {double re; double im;};
void add_on( struct complex *first , struct complex *second);
    /* add_on function adds two complex
                numbers and assigns the result to *first
    */
int main(){
    struct complex x,y;
    x.re=2.4; y.re=4.0;
    x.im=3; y.im=6;

    printf("x is %f + %fi\n", x.re, x.im);
    printf("y is %f + %fi\n", y.re, y.im);
    add_on(&x,&y);
    printf("x is %f + %fi\n", x.re, x.im);
    printf("y is %f + %fi\n", y.re, y.im);        return 0; }

void add_on( struct complex *a , struct complex *b)
{
    (*a).re= (*a).re +(*b).re;   // OR    a->re= a->re + b->re;
    (*a).im= (*a).im +(*b).im; // OR    a->im= a->im + b->im;
    // return ;
}
```

ACCESSING MEMBERS OF A STRUCTURE

```
#include<stdio.h>

struct student{
       int stID;
       char lastName[25];
       char grade;
};

int main(){
      struct student young;
       struct student *ptr=&young;
      young.stID=201406;
       scanf("%s", young.lastName);
       young.grade='A';    // ptr->grade= 'A';

       printf("%d\n",young.stID);
      printf("%d\n",ptr->stID);
       printf("%d\n",(*ptr).stID);

      printf("%c\n",young.grade);
      printf("%c\n",ptr->grade);
       printf("%c\n",(*ptr).grade);

      printf("%s\n",young.lastName);
      printf("%s\n",ptr->lastName);
      printf("%s\n",(*ptr).lastName);
       return 0;
}
```